A Hybrid Question Answering System based on Ontology and Topic Modeling

Kwong Seng Fong, Chih How Bong
Faculty of Computer Science and Information Technology,
Universiti Malaysia Sarawak, 94300, Kota Samarahan, Sarawak, Malaysia.
ksfong@siswa.unimas.my

***Abstract*—A Question Answering (QA) system is an application which could provide accurate answer in response to the natural language questions. However, some QA systems have their weaknesses, especially for the QA system built based on Knowledge-based approach. It requires to pre-define various triple patterns in order to solve different question types. The ultimate goal of this paper is to propose an automated QA system using a hybrid approach, a combination of the knowledge-based and text-based approaches. Our approach only requires two SPARQLs to retrieve the candidate answers from the ontology without defining any question pattern, and then uses the Topic Model to find the most related candidate answers as the answers. We also investigate and evaluate different language models (unigram and bigram). Our results have shown that this proposed QA system is able to perform beyond the random baseline and solve up to 44 out of 80 questions with Mean Reciprocal Rank (MRR) of 38.73% using bigram LDA.**

***Index Terms*—Knowledge-Based Approach; Language Model; QA System; Text-Based Approach**

I. Introduction

A Question Answering (QA) system is an application that can provide accurate answers to the user's natural language questions. In recent years, the demand for automated QA system becomes very high. It is because it is a suitable learning platform for active and unsupervised learning. The students can seek help from QA systems when they have questions. It will be helpful if there are automated QA systems, which assist the students to learn a subject effectively and efficiently. For example, "*Ask.com*", "*Yahoo! Answers*" and "*START Natural Language QA System*", they all focus on multiple English topics. The users can input the question to the system in order to obtain the answers and information.

There are works using various approaches to improve the performance of the QA systems. For knowledge-based approach, it uses various predefined templates to form triple patterns in order to solve different question types. Sometimes, it tries to form complex triple patterns, so that it can select the appropriate answers. For text-based approach, it uses different Information Retrieval (IR) to find the answers.

In this paper, we proposed a hybrid QA system, which is a combination of both knowledge and text-based approaches. It can solve five types of the questions: factoid types, description types, definition types, reason types and relation types. Overall, our intention is to remove the complication of defining patterns and to improve the answer retrieval.

In the following section, we will present two approaches to QA systems. Section 3 will explain Latent Dirichlet Allocation (LDA). Section 4 will discuss the Q&A system architecture. Section 5 will describe a Physics ontology to support QA. Section 6 will detail out our QA system architecture. Section 7 will present the results of the QA system. Finally, section 8 will present the paper conclusion and future works.

II. Approaches of QA Systems

In general, there are two types of QA systems: knowledge-based and text-based approaches.

A. Knowledge-based Approach

The knowledge-based approach has a structured knowledge base (KB). The approach embeds the keywords of the question into predefined templates to form triple patterns, which is a semantic representation of the questions to be used to extract the answers from the KB [1].

However, this approach has two factors that may affect the system performance: forming the triple pattern and retrieving the answers from the KB [1]. This is because if the system cannot form the triple pattern correctly, it cannot retrieve the correct answers from the KB. The formation of the triple pattern can be very complicated as different question types require different templates. For example, the Boolean question require the template of "*ASK WHERE ?x ?p ?y*" and the simple question require the template of "*SELECT DISTINCT ?x WHERE ?x ?p ?y*" where *?x*, *?p* and *?y* are proxy variables [2].

B. Text-based Approach

The text-based approach is also known as the IR approach. This approach retrieves information from a text collection, where is unstructured. The overall process of the approach is converting the question into a query, which is a list of the keywords, and then input the query into the IR or search engine to find the relevant documents (also answers) [3][4]. With a list of relevant documents, the system ranks the most relevant documents as the answers to the questions.

However, this approach also has two factors which may affect the system performance: the formation of the query from the question and ranking of the relevant documents. It is because if the question cannot be converted into meaningful query, the system cannot find the most relevant answers, hence yielding low precision and recall [5].

III. Latent Dirichlet Allocation (LDA)

LDA is a generative probabilistic model for a set of discrete data likes text corpora [6]. It presents the documents as a

mixture of many topics, where each topic is characterized by the words distribution [6]. Figure 1 displays the graphical model of LDA.

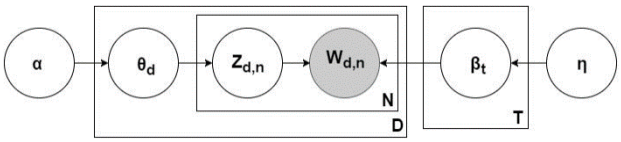

Figure 1: A graphical model representation LDA [7]

With LDA, every document is represented as a random mixture of the latent (hidden) topics and each latent topic is represented by a distribution over vocabulary that exists in the document.

For each topic T, it draws a distribution over vocabulary β_t based on the multinomial distribution with the Dirichlet parameters η. For each document D in the corpus, it draws a distribution over topics θ_d based on the multinomial distribution with the Dirichlet parameters α. Then, for the nth word in the document d, it draws a topic $Z_{d,n}$ based on the multinomial distribution with the parameter θ_d, where $Z_{d,n} \in \{1, \ldots, T\}$ and draws the observed word $W_{d,n}$ based on the multinomial distribution with parameters $\beta_{Zd,n} \in \{1, \ldots, V\}$ and V is the vocabulary size.

Given the observed words in a set of documents, we analyze them by computing the posterior distribution of the hidden variables (Z, θ, β). Hence, the main purpose of LDA is to infer the posterior distribution of the latent topic variables after observing the training data.

$$p(\theta, Z \mid w, \alpha, \beta) = \frac{p(\theta, Z, w \mid \alpha, \beta)}{p(w \mid \alpha, \beta)} \quad (1)$$

However, computing this distribution is intractable, we must find another approach to infer the hidden variables. One of the approaches to estimate posterior inference is Variation Bayes (VB) approach [8]. In VB inference, the true posterior distribution is approximated by a simpler and full factorized distribution q and associated parameters φ, γ, λ with the original parameters Z, θ, β respectively. We select $q(Z, \theta, \beta)$ of the form $q(Z_{d,n}=T) = \varphi_d, W_{d,n}T, q(\theta_d) = Dirichlet(\theta_d, \gamma_d)$ and $q(\beta_t) = Dirichlet(\beta_t, \lambda_t)$ where the posterior over the per-word topic assignments Z is parameterized by φ, the posterior over the topics β is parameterized by λ. Our goal is to estimate φ, γ, λ by using Expectation Maximization (EM) method to alternate between two steps: (1) E-step estimates φ and γ by using the current λ value and (2) M-step updates λ by using the current φ value.

The E-step only uses the current chunk where the chunk can be a single document, some if documents or the whole document collection. In this step, φ and γ are iteratively updated by using the Equation (2) and Equation (3) until convergence in order to find locally optimal values for φ and γ where λ is fixed holding.

$$\varphi_{iter,w,T} \propto \exp\{E_p[\log \theta_{iter,T}] + E_q[\log \beta_{T,w}]\} \quad (2)$$

$$\gamma_{iter,T} = \alpha + \sum_w n_{iter,w} \varphi_{iter,w,T} \quad (3)$$

where: $n_{iter,w}$ = Occurrence number of words, w, in the current iteration's chunk of documents.

In the M-step, λ' is computed, which is the λ value if the whole document collection is made up of (number of documents/chunk size) copies of the current chunk. Then, λ is updated by using a weighted average of its previous value and λ' as shown in Equation (4) and (5).

$$\lambda'_{T,w} = \eta + batchSize * n_{iter,w} \varphi_{iter,w,T} \quad (4)$$

$$\lambda = (1 - \rho_{iter})\lambda + \rho_{iter}\lambda' \quad (5)$$

where: ρ_{iter} = Weighted parameter.

After λ is estimated, we can find the most possible words for each topic by looking at the word probabilities in each row of λ.

A simple example below shows how to determine the similarity of the documents to the query "*meaning of the first law of thermodynamics*" by using LDA. Assume that there have three documents in the document collections that are used to learn topic models as shown in Table 1.

Table 1
The Documents Used to Learn Topics Models with the LDA

Document	Text
D1	The electric charge is a fundamental conserved property of certain subatomic particles that determines their electromagnetic interaction. Electrically charged particles are influenced by and create …
D2	The first Law of Thermodynamics states that heat is a form of energy, and thermodynamic processes are therefore subject to the principle of conversation of energy. This means that heat energy cannot …
D3	Each electron in an atom has an orbital magnetic dipole moment and a spin magnetic dipole moment. The resultant of these two vectors combines with similar resultants for all other electrons in …

Then, given T topics (where T = 3), the LDA model automatically learns those topics and assign them to the documents. The learned topics are represented by the words and their existing probabilities in every topic. Table 2 shows the first three words of the highest probability in each topic. Table 3 shows the probability distribution over the topics for every document.

Table 2
First Three Words of the Highest Probability in Each Topic

Topic 1	
Word	Probability
electromagnetic	0.056
energy	0.051
magnetic	0.035
Topic 2	
Word	Probability
energy	0.058
heat	0.046
magnetic	0.031
Topic 3	
Word	Probability
magnetic	0.065
electromagnetism	0.048
atom	0.039

Table 3
The LDA Topic Assignments to the Documents and the Query

	Topic 1	Topic 2	Topic 3
D1	0.5688	0.0903	0.3408
D2	0.0923	0.8137	0.0940
D3	0.0958	0.0952	0.8090
Query	0.1559	0.6893	0.1548

$$similarity = \cos(\theta) = \frac{d_i . q}{\| d_i \| \| q \|} \quad (6)$$

where: d_i = A set of topic distribution for document i
q = A set of topic distribution for the query

Then, we use the Equation (6) to calculate the cosine similarity between the probability distribution over the topics of each of the documents and the query as shown in Table 4. Given the similarity score is x, where $-1.0 \leq x \leq 1.0$. If the similarity score is 1.0, it indicates an exact match between the query and the document. In contrast, if the similarity score is -1.0, it means they are totally unmatched, In other words, a higher similarity score indicates a higher relatedness of the document to the query. Based on the example given above, the documents will be ranked in the following: D2, D1 and D3 (see Table 4).

Table 4
Cosine Similarity Scores Between the Query and the Documents

(Query and documents)	Cosine similarity score
(Query, D1)	0.4207
(Query, D2)	0.9891
(Query, D3)	0.3468

IV. Related Works

In this section, we describe four existing QA systems that relevant to our proposed QA system. According to the Kamdi and Agrawal, they proposed a QA system for Indian Penal Code (IPC) and Indian Laws by using machine learning method and IR approaches [9].

In addition, Cui and Wang presented a LDA QA system on database principle [10]. The system analyses the question, which included word segmentation, question classification, keywords extraction and keywords expansion. Then, it uses the LDA model to identify and retrieve a set of related predefined questions, which corresponding with their answers, from the database. Next, cosine similarity is used to determine the similarity scores.

Moreover, Abdi, Idris and Ahmad have developed an ontology-based QA system for the Physics domain (QAPD) [11]. First of all, the questions were preprocessed -- tokenization, part-of-speech tagging, stemming, stopwords removal and annotation, to form a query. Then, the system uses the inferring schema mapping (ISM) method, which is the combination of semantic and syntactic information as well as attribute-based inference, to calculate the ISM coefficient score between the query and the predefined set of the query patterns, which are retrieved from the query database to select a suitable query pattern and the Structured Query Language (SQL) statement that can extract the expected answer from the ontology, which is a KB for Physics concepts.

We found out that most of the QA systems have the same basic modules. For the text-based QA system, after it analyzes, preprocesses and classifies the question, it performs the IR approach and filtration to extract the candidate answers from the document collections. Then, the system identifies the answers from the candidate answers in the form of words, phrases or sentences. For the knowledge-based QA system, the overall processes are slightly different from the text-based QA system. After the question is preprocessed, the system forms the SQL commands to extract the answer from the ontology. However, as mentioned before, this kind of QA system is heavily depending on the formation of the SQL commands, which require many different predefined templates for different question types.

Therefore, in this paper, we propose a QA system using a hybrid approach, which is a combination of the knowledge-based and text-based approaches. The proposed QA system just only uses two Simple Protocol and RDF Query Languages (SPRAQLs), without the query patterns restriction, to determine the candidate answers from the ontology. Then, the system uses LDA to extract the answers from the candidate answers. For the sake of brevity, the proposed QA system will try to solve five types of the questions: definition, reason, relation, description and factoid type.

V. Using Ontology to Support QA

Ontology is a centralized repository for information. It is used to capture knowledge and describe the concepts in the domain and the relationships between the concepts [12]. It has several components which are summarized in Table 5.

Table 5
Components of the Ontology

Component	Explanation
Classes	They are the sets that consist of instances.
Instances	They are referred to as the individuals or objects in the domain that we are interested.
Properties	
Object properties	They show the relationship between two instances.
Data properties	They show the relationship between an instance and its data (datatype) values.
Data values	They are literal which not being treated as instances.

Since the proposed QA system can be applied to many domains, we take the secondary school Physics subject as our testing and evaluation platform. The collected data is extracted from six e-books, which is in the form of paragraph. However, the paragraphs are lengthy and contain too much information which are unsuitable to be the returned answers since the returned answer for the QA system must be simple and short. Sentences are preferred since they can provide more textual information compared to the words or phrases [12]. Therefore, the returned answers for our proposed QA system are in the form of sentences as shown in Figure 2.

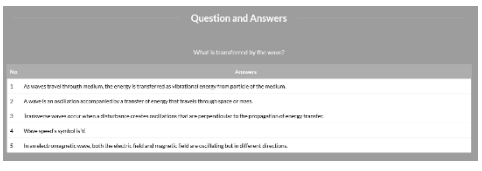

Figure 2: The interface of the proposed QA system

We intend to facilitate the ontology as the KB to provide the possible sets of candidate answers, which are in the form of sentences. The ontology is known as Physics Concepts Ontology (PCO). We use protégé to construct the ontology. It is a free, open-source ontology editor and framework [13]. During the ontology construction, the collected data can be categorized into four classes in the PCO as shown in Table 6.

Table 6
The Four Classes in the PCO

Class	Explanation
Constant value term	Store constant values such as absolute zero temperature, gas constant and specific heat of water.
Device term	Store devices such as ammeter, calorimeter and voltmeter.
Physics term Electricity term Electromagnetism term Thermodynamics term Waves term	Store instances about the terms that do not belong to other classes such as electric field, amplitude and magnetism.
Unit term	Store the units such as ampere, joule and coulomb.

For the data collection, the entities are stored as the instances, the information of the entities are stored as the data values, the relationships between the information and the entities are stored as the data properties as well as the relationships between two entities are stored as the object properties. For example, "*Electric power is the rate at which electric energy is transferred by an electric circuit. The SI unit of power is the watt.*" Its instance, data property with data value and object property are stored as in Table 7.

Table 7
An Example that the Values are Stored in the Components

Component	Value
Instances	Electric power
Data property	Definition
Data value	Electric power is the rate at which electric energy is transferred by an electric circuit
Object property	Has unit of (links to "*watt*")

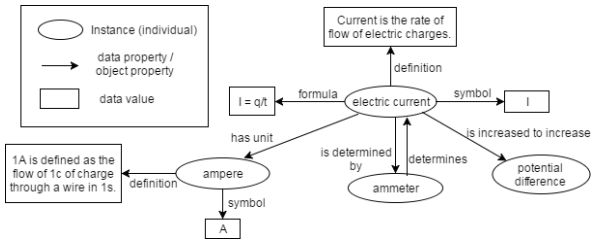

Figure 3: The interconnection between four instances in the PCO

Figure 3 shows an example of the interconnection between four instances, which are electric current, ammeter, potential difference and ampere, in the PCO. In this figure, the "electric current" has three data properties with data values and three object properties, which links to other instances as in Table 8.

Our approach uses the SPARQLs to retrieve the returned results from the PCO as the candidate answers in the Candidate Answers Retrieval (CAR) module. The returned results of the PCO is expected in the form of sentences, but sometimes it may returns the results in the form of words or phrases, especially for those SPARQLs that use the object properties. The module itself needs to combine the query's keyword, the alternative name of the property (data property or object property) and the returned result (words or phrases) to form a complete sentence as a candidate answer. The alternative name is the predefined name of the property that can suitably be used in the sentence. Table 9 shows four examples of the data properties and four examples of the object properties corresponding with their alternative name those are stored in the Property Text File (PROTF). Table 10 shows an example of the sentence formation.

Table 8
The Data Properties and the Object Properties for "*Electric Current*" Instance

Data property	Data value
formula	I = q/t
definition	Current is the rate of flow of electric charges.
symbol	I
Object property	**Another instance**
has_unit	ampere
is_determined_by	ammeter
is_increased_to_increase	Potential difference

Table 9
Examples of the Data Properties and the Object Properties Corresponding with Their Alternative Name

Data property	Alternative name
definition	""
characteristic	""
constant_value	"*'s constant value is* "
formula	"*'s formula:* "
Object property	**Alternative name**
has_unit	"*'s unit is* "
is_used_in	" *is used in* "
no_depend	" *does no depends on* "
equals_to	" *equal to* "

Table 10
An Example of the Sentence Formation

Component	Value
Instances	electric current
Object property	has_unit
Alternative name	"*'s unit is* "
Returned result	ampere
Object property	Electric current's unit is ampere

VI. The Proposed QA System Architecture

Our QA system has five modules, which are (1) Data Modeling (DM), (2) Ontology Construction (OC), (3) Question Preprocessing (QP), (4) Candidate Answers Retrieval (CAR) and (5) Answer Extraction (AE) as shown in Figure 4.

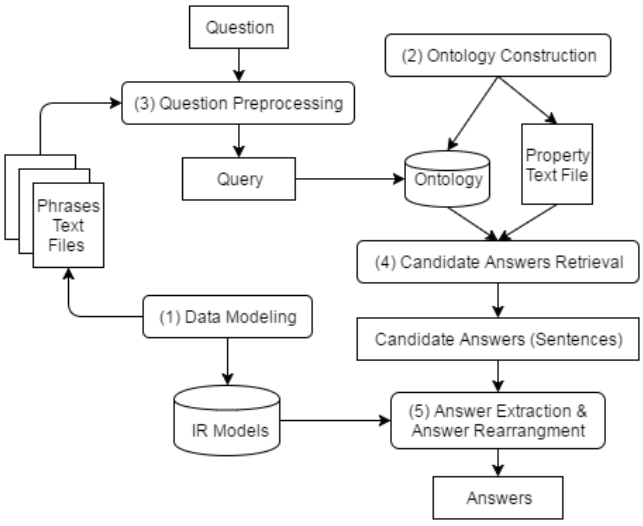

Figure 4: The architecture of the proposed QA system

A. Data Modeling (DM)

DM is used to construct semantic KBs which are used to discover semantic similarity between the query and the candidate answers. The input of DM is the paragraphs

Physics learning units. All paragraphs are preprocessed through the preliminary processes, which are lowercase conversion, tokenization, abbreviation expansion, stopwords removal, stemming and n-gram determination.

Lowercase conversion is the process to convert the sentence into lowercase letters to increase recall. Given a sentence from the paragraph as an example, "*The first law of thermodynamics states that energy is conserved*.", after the lowercase conversion, the output is "*the first law of thermodynamics states that energy is conserved*."

Tokenization is the process to split the sentence into a sequence of token (words). At the same time, all the punctuation of the sentence is removed. As a result, the preprocessed sentence will be "*['the', 'first', 'law', 'of', 'thermodynamics', 'states', 'that', 'energy', 'is', 'conserved']*".

Abbreviation expansion is the process to resolve abbreviations. For example, "*emf*" is abbreviation of the "*electromotive force*".

Stopwords removal is the process to remove insignificant words by using the stopwords list in Natural Language Toolkit (NLTK), such as "what", "is" and "the". After the stopwords removal, the preprocessed sentence will be "*['first', 'law', 'thermodynamics', 'states', 'energy', 'conserved']*".

Stemming is the process to reduce the words to their stem form. The preprocessed sentence will be "*['first', 'law', 'thermodynam', 'state', 'energi', 'conserv']*".

N-gram determination is the process to form the words of the sentence in unigram or bigram. In our context, the meaning of unigram and bigram are slightly different than the original meaning. Unigram is a word sequence of words likes "*first*", "*law*" and "*thermodynam*". Bigram is a two-word sequence of words likes "*first_law*" or "*law_thermodynam*" with unigram. For example, after the bigram determination, the preprocessed sentence will be "*['first', 'law', 'thermodynam', 'state', 'energi', 'conserv', 'first_law', 'law_thermodynam', 'thermodynam_state', 'state_energi', 'energi_conserv']*".

After the preprocessing, all the words, which have more than five occurrence frequency, are stored into the Phrases Text Files (PTF) with their occurrence frequency. Words for unigram paragraphs are stored into the unigram PTF. Different n-grams are stored respectively.

Preprocessed paragraphs are then used to construct two LDA models, each model in different language models (unigram and bigram) in order to examine their efficacy. See Section 3 for instructions to build the LDA model. For unigram LDA, it uses the VB to generate the word-topic probabilities and per-paragraph topic distribution in order to infer the topic distribution for the query and the candidate answers. Likewise, bigram LDA model is also constructed.

B. Ontology Construction (OC)

OC is used to construct an ontology, which is named as Physics Concepts Ontology (PCO), to provide domain specific knowledge and support conceptualized explanations. Section 5 details out how to build the ontology from the paragraphs. Once we have built the ontology, it will be used as the repository to retrieve the candidate answers to the question in the CAR module.

C. Question Preprocessing (QP)

QP is used to transform a question into a query, which is a set of keywords. The question is preprocessed with lowercase conversion, tokenization, abbreviation expansion, stopwords removal, stemming and n-gram determination to form a query. For the proposed QA system, it solves five types of the questions: definition, reason, relation, description and factoid type as shown in Table 11. After the query is formed, it will be passed to the CAR module for further processing.

D. Candidate Answers Retrieval (CAR)

CAR is used to retrieve all possible candidate answers with the aid of PCO. Two SPARQLs are needed to seek the candidate answers. SPARQL is a query language for Resource Description Framework (RDF) which is used to retrieve information from the KB [14].

The module uses the first SPARQL to retrieve the instances from the PCO. The query's keywords will be the input and automatically embed into the predefined SPARQL statement (7) in order to form the first SPARQL.

$$\begin{aligned}&SELECT\ ?individual\ WHERE\{\\&?individual\ rdf:type\ owl:NamedIndividual.\\&FILTER(\\&regex(str(?individual),'\#"+queryKeyword+"')\|\|\\&(regex(str(?individual),'\_"+queryKeyword+"'))\\&)\}\end{aligned} \tag{7}$$

where: *?individual* = The instance we are interested in

From this SPARQL statement, it retrieves the instances or its substring that fulfill the filtration requirement. For example, given a bigram query that consists of five keywords, "*['definit', 'emf', 'electromot', 'forc']*", the first keyword "definit" is embedded into the SPARQL statement (7) in order to form the SPARQL (8).

$$\begin{aligned}&SELECT\ ?individual\ WHERE\{\\&?individual\ rdf:type\ owl:NamedIndividual.\\&FILTER(\\&regex(str(?individual),'\#definit')\|\|\\&(regex(str(?individual),'\_definit'))\\&)\}\end{aligned} \tag{8}$$

The rest of the four keywords are also used to form another four SPARQLs. With these SPARQLs, the module retrieves the instances from the PCO. The returned instances are "*force*", "*electromotive_force*" and "*ampere's_force_law*".

Then, the module automatically embeds each of the instance into the predefined SPARQL statement (9) with the property (data property or object property), which is extracted from the PROTF, to form the second SPRAQL in order to retrieve the related sentences from the PCO as the candidate answers.

$$\begin{aligned}&SELECT\ ?object\ WHERE\{\\&?subject\ \ foo:"+property+"?object\},\\&initBinding=\{"subject":ins\tan ce\}\end{aligned} \tag{9}$$

where:
foo = Path that connects to the PCO
property = Name of the property which is extracted from the PROTF

?subject = The instances where it is assigned by those returned instances with the initial binding function

?object = The returned result, which is in the form of word, phrase or sentence

For example, the instance "*electromotive_force*" is embedded into the SPARQL statement (9) with the data property "*definition*" in order to form the SPARQL (10).

$$\begin{aligned} &SELECT\ ?object\ WHERE\{ \\ &?subject\ \ foo:"+property+"?object\}, \\ &initBinding = \{"subject": electromotive\_force\} \end{aligned} \quad (10)$$

The returned result is "*The electromotive force (e) or e.m.f. is the energy provided by a cell or battery per coulomb of charge passing through it, it is measured in volts (V).*" The module depends on the alternative name of the property, which have been pre-stored in the PROTF, to determine the return form of the result. If there is no value for the alternative name, the returned result is a sentence and is considered as the candidate answer. If there has an alternative name in the PROTF, the returned result is either a word or a phrase. The module will combine the instance, the alternative name of the property and the returned result to form a sentence as the candidate answer. However, if there is no value for the alternative name of the data property "*definition*", the module considers the returned result is in the form of sentence, which is also a candidate answer.

If the instance "*electromotive_force*" is embedded with the object property "*has_unit*", the returned result is "*volt*". In the PROTF, the alternative name of the object property "*has_unit*" is "*'s unit is* ", so the returned result is a word or a phrase. The module needs to form a complete sentence by combining them. The sentence is "*Electromotive force's unit is volt*." It is another candidate answer.

Another two instances "*force*" and "*ampere's force_law*" are also assigned into the SPARQL statement (9) to form SPARQLs to retrieve the related sentences as candidate answers. Those candidate answers are then further processed in the AE module.

E. Answer Extraction (AE)

AE is used to rank the candidate answers and produce the best five answers. Given a bigram query "*['defin', 'heat', 'capac', 'heat_capac']*", after a list of candidate answers is retrieved from the PCO, the module uses bigram LDA model to find the relatedness between the query and the candidate answers. Firstly, the module converts the query and the candidate answers into the probability distribution over the topics. Then, it calculates the cosine similarity between the probability distribution over the topics of the query and the candidate answers. Based on the similarity score, it ranks the best five candidate answers. Figure 5 shows an example of how to produce the ranked candidate answers by using the bigram LDA.

In general, the process of the LDA model in the unigram is also identical. The differences between them are the n-gram query that is used as the input and the n-gram trained LDA model being used.

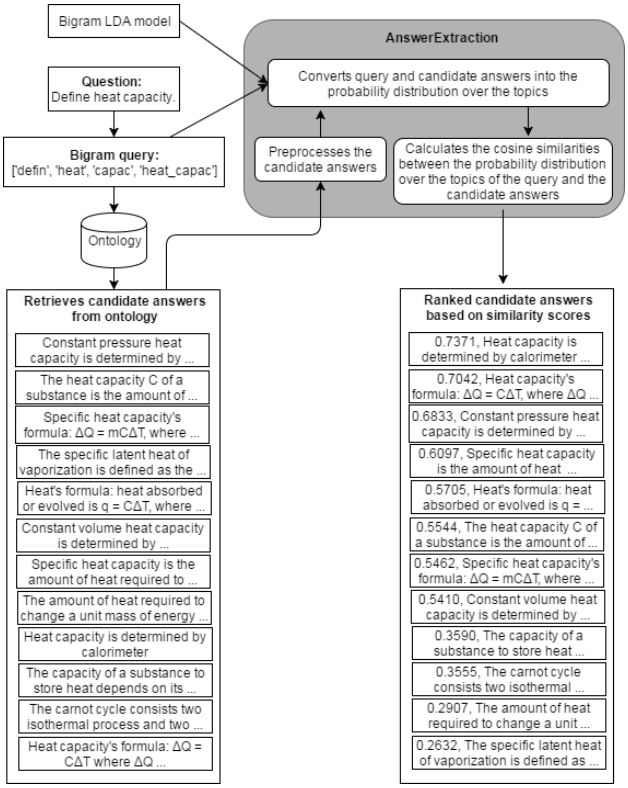

Figure 5: An example to produce the ranked candidate answers by using bigram LDA

VII. EXPERIMENTS

A. Gold Standard

We use 80 questions with answers (20 questions per categories) from the textbooks as the gold standard to evaluate the proposed QA system. The average length for every question is around 10 words. The questions can be any of the five types shown in Table 11. The table also shows the total number of the questions in the gold standard.

Table 11
The Total Number of Questions and One Example of Every Question Type

Question type	Number of questions	Example
Definition	27	Give the meaning of power.
Reason	4	State why alternating current power supply is used.
Relation	11	What is the relationship between power and energy?
Description	14	Explain the difference between a longitudinal wave and a transverse wave.
Factoid	24	What is the SI unit for electric charge?
Total	80	

B. Evaluation Measures

We use Top Five Accuracy (TFA) [15] and the Mean Reciprocal Rank (MRR) [16] to evaluate the performance of the proposed QA system.

The equation for the TFA is given as the following:

$$TFA = \frac{N_{answered\_question}}{N} \quad (11)$$

where: $N_{answered\_question}$ = Total number of the questions that the system returns the correct answer in the top five answers

and N = Total number of questions that are evaluated by the system.

The equation for the MRR is given as the following:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i} \quad (12)$$

where: N = Total number of questions that are evaluated by the system and $rank_i$ = The highest rank position of the answer for the ith question

C. Parameters Estimation for LDA

Here, we intent to find the most likely per-document topic distribution and the most likely topic distribution. In the experiments, we use η = 0.01 (prior on the words of given topic) and α = 50/number if topics (prior on topics of a given document) because these parameters are reported working well in the past literature [17][18]. We also set 100 iterations for each inference [19]. However, we have to determine the most suitable number of topics T. One of the solutions is to directly evaluate the proposed QA system by using LDA with different number of topics T in order to determine the values contributed to the highest TFA and MRR shown in Table 12.

Table 12
Finding the Best T for the N-gram

	LDA_5 (unigram)		LDA_5 (bigram)	
T	TFA	MRR	TFA	MRR
50	0.4375	0.2408	0.4750	0.3090
100	0.4750	0.2917	***0.5500***	***0.3873***
150	0.5125	0.2873	0.4500	0.2910
200	***0.4875***	***0.2950***	0.4625	0.2788
250	0.4375	0.2477	0.4500	0.2869
300	0.4500	0.2363	0.5000	0.3021
350	0.4750	0.2786	0.4875	0.2910
400	0.4500	0.2890	0.4375	0.2785
450	0.4000	0.2706	0.4875	0.3000
500	0.4250	0.2385	0.4125	0.2694

In our context, we believe that MRR carries more weight than TFA since it can determine the average of the correct answers and the average precision of the correct answers over 80 questions. For unigram, we set T = 200. For bigram, we set T = 100.

In addition, the baseline of the comparison is against the random selection, $Rand_5$ [20, 21]. In the experiment, we set five iterations for the random selection. For each iteration, the system randomly selects an answer from the candidate answers. For the first random selection, the first selected candidate answer will be the first returned answer. After the five iterations, the five answers are selected. Besides that, we try to test the LDA model with two different language models (unigram and bigram) in order to determine which has better performance.

D. Experimental Results

Table 13 shows both the TFA and MRR of the random baseline and LDA model. Overall, all the LDA models perform better than the random baseline. The best performer is the bigram LDA_5, which yields a TFA of 55% (44 correct answers out of 80 questions) with a MRR of 38.73%. It followed by the unigram LDA_5, which yields a TFA of 48.75% (39 correct answers out of 80 questions) with a MRR of 29.50%.

Table 13
Results for the N-gram

	Unigram		Bigram	
	TFA	MRR	TFA	MRR
$Rand_5$	0.3500	0.2310	0.3500	0.1910
LDA_5	0.4875	0.2950	0.5500	0.3873

From the results, it is obviously to show that LDA_5 in bigram performs better than unigram. It is because bigram is able to regard "*electric current*", "*electromotive wave*" and "*heat capacity*" as significant words. Separating those words will render their meaning differently.

E. Why the System cannot return the Correct Answers?

Some of the question cannot be answered correctly because no information is encoded in the ontology, PCO. Table 14 shows a question that the system could not retrieve the most related candidate answers from the PCO because no relevant information is stored in the PCO.

Table 14
A Question that the System Cannot Answer Correctly

Question	What precaution should you take when taking reading from an ammeter or a voltmeter?
Expected answer	Avoid zero error in meters.

Another reason is some of the questions may have more than one answer, however, there is only one answer to each gold standard question. When the returned answer does not match with the expected one, the system renders them incorrectly. Table 15 shows an example. From this example, the returned answer, likes "*Resistance is increased to decrease electric current*" or "*Resistance is inversely proportional cross-sectional area*", are different from the expected answer, so they are considered as incorrect.

Table 15
A Question that Can Have More Than One Answer

Question	Suggest one way to increase the resistance of the ohmic conductor.
Expected answer	Increase the length of the ohmic conductor

F. Limitations

The proposed QA system has two weaknesses. The first weakness is we only have limited resources to build the models and it may cause the system retrieves limited candidate answers from the ontology. It also reduces the accuracy of the system to answer the question. The second weakness is the proposed QA system is able to solve five question. For example, the system cannot solve the list type and Boolean type. For list type, it cannot list out all the possible answers as one answer. For Boolean type, it cannot answer "*yes*" or "*no*", as it will return a sentence as the answer.

VIII. Conclusion and Future Works

We proposed a hybrid QA system, which is a combination of the knowledge and text-based approaches. It only required two SPARQLs to retrieve the candidate answers from the ontology without the question templates. Our experiment results showed that the proposed QA system has performed well. The bigram LDA_5 produces the best result, which is better than unigram.

There are several works that can be implemented in the QA system for better performance. The system can be further enhanced by increasing the information content in the ontology as more data will increase the precision and recall. In addition, the system can also be augmented to present the answers in the form of images or videos along with sentences to better help the users to understand the answers easily. This is very true especially for the reason and description type questions.

ACKNOWLEDGMENT

We would like to thank Universiti Malaysia Sarawak (UNIMAS) and KPT who funded this study through the RACE/b(6)/1098/2013(06).

REFERENCES

[1] J. Bao, N. Duan, M. Zhou, and T. Zhao, "Knowledge-based question answering as machine translation," *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, vol. 2, no. 6, pp. 967-976, Jun. 2014.

[2] S. Park, H. Shim, and G. G. Lee, "Isoft at qald-4: Semantic similarity-based question answering system over linked data," in *Clef (Working Notes)*, 2014, pp. 1236-1248.

[3] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. New Jersey, United States: Prentice Hall, 2015.

[4] J. Zhou, H. Zhang, and D. Lo, "Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports," in *2012 34th International Conference on Software Engineering (ICSE)*, 2012, pp. 14-24.

[5] P. Gupta and V. Gupta, "A survey of text question answering techniques," *International Journal of Computer Applications*, vol. 53, no. 4, pp. 1-8, Jan. 2012.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation, " *Journal of Machine Learning Research*, vol. 3, pp. 993-1022, Jan. 2003.

[7] H. B. Jeon and S. Y. Lee, "Language model adaptation based on topic probability of latent dirichlet allocation," *ETRI Journal*, vol. 38, no. 3, pp. 487-493, Jun. 2016.

[8] M. D. Hoffman, D. M. Blei, and F. Bach, "Online learning for latent dirichlet allocation," in *Advances in Neural Information Processing Systems*, 2010, pp. 856-864.

[9] R. P. Kamdi and A. J. Agrawal, "Keywords based closed domain question answering system for indian penal code sections and indian amendment laws," *International Journal of Intelligent Systems and Applications*, vol. 7, no. 12, pp. 57-67, Nov. 2015.

[10] L. Cui and C. Wang, "An intelligent q&a system based on the lda topic model for the teaching of database principles," *World Translations on Engineering and Technology Education (WIETE)*, vol. 12, no. 1, pp. 26-30, 2014.

[11] A. Abdi, N. Idris, and Z. Ahmad, "Qapd: an ontology-based question answering system in the physics domain," in *Soft Computing*, 2016, pp. 1-18.

[12] M. Horridge and S. Brandt, *A Practical Guide to Building Owl Ontologies Using Protégé 4 and Co-ode Tools Edition 1.3*. Manchester, United Kingdom: The University of Manchester.

[13] T. Tudorache, J. Vendetti, and N. F. Noy, "Web-protégé – protégé going web," in *Stanford Center for Biomedical Informatics Research*. California, United States: Stanford University, 2008.

[14] E. Prud'hommeaux and A. Seaborne, *SPARQL Query Language for RDF*. W3C Recommendation 15 January 2008, https://www.w3.org/TR/rdf-sparql-query/.

[15] K. Komiya, Y. Abe, H. Morita, and Y. Kotani, "Question answering system using q & a site corpus query expansion and answer candidate evaluation," *SpringerPlus*, vol. 2, no. 396, pp. 1-11, Dec. 2013.

[16] N. Craswell, "Mean reciprocal rank," in *Encyclopedia of Database Systems*, S. Lappin and C. Fox, Ed. Malden, Massachusetts: Wiley-Blackwell, 2013.

[17] A. Asuncion, M. Welling, P. Smyth, and Y. W. The, "On smoothing and inference for topic models," in *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 2009, pp. 27-34.

[18] M. Steyvers and T. Griffiths, "Probabilistic topic models," in *Handbook of Latent Semantic Analysis*, vol. 427, T. K. Landauer, D. S. McNamara, D. Simon, and W. Kintsh, Ed. New Jersey, United States: Lawrence Erlbaum Associates, 2007, pp. 424-440.

[19] I. Sato and H. Nakagawa, "Rethinking collapsed variational bayes inference for lda," in *Proceedings of the 29th International Conference on Machine Learning*, 2012, pp. 999-1006.

[20] M. Light, G. S. Mann, E. Riloff, and E. Breck, "Analyses for elucidating current question answering technology," *Natural Language Engineering*, vol. 1, no. 4, pp. 325-342, Dec. 2001.

[21] D. W. Oard, J. White, J. Paik, R. Sankepally, and A. Jansen, "The fire 2013 question answering for the spoken web task," in Post-Proceedings of the 4th and 5th Workshops of the Forum for Information Retrieval Evaluati, 2013, pp. 1-8.